Travaux pratiques n° 5 : Contrôle de trafic dans les routeurs

09/04/2014

A l'issue de ce TME, vous devrez envoyer un compte rendu à l'adresse `pierre.tournoux@univ-reunion.fr`. Son l'évaluation comptera pour le CC.

Ce TME a pour objectif de montrer le rôle des routeurs dans la performance de TCP. Nous allons voir en premier les problèmes occasionnés à TCP par une gestion de file d'attente inappropriée. Ensuite nous verrons les apports d'une gestion active des paquets par les routeurs. Au préalable vous devez récupérer l'archive contenant les fichiers nécessaires à la réalisation du TME à l'adresse : `http://www.pu-tournoux.net/` » Teaching » Master 1 RES - UPMC.

Exercice 1 — Défauts de Droptail

Dans cet exercice nous allons étudier comment interagissent le réseau et les flots TCP ainsi que le rôle de la discipline d'attente (*queue management*) dans la performance de TCP. Nous mettrons en évidence les principaux défauts de la discipline d'attente dite "Droptail". Pour réaliser les mesures demandées, vous utiliserez le simulateur réseau `ns-2`[1]. L'annexe 1.1 décrit l'environnement de simulation et les scripts à utiliser. Lisez la attentivement. La figure 1 représente la topologie du réseau que nous utiliserons pour cette étude. Ses caractéristiques sont :

- le tampon mémoire pour le lien (R_1, R_2) est de 22 paquets (`queue-limit $r1 $r2 22`).
- la version de TCP utilisée est newreno (`Agent/TCP/Newreno`).
- la fenêtre de contrôle de flux des connexions est limitée à 60 segments TCP (`set window_ 60` *c.a.d.* 60 paquets). Cela signifie que la fenêtre de congestion (`cwnd`) sera d'une taille inférieur ou égale à 60 segments.
- la durée de la simulation est de 50 secondes.
- la taille des paquets est de 1000 octets (`set packetSize_ 1000`).

Il y a deux flots dans le réseaux : flot 1 émis par le noeud S_1 à destination de D_1 et le flot 2 émis par le noeud S_2 à destination de D_2. Le RTT rencontré par le flot 1 est noté RTT_1 et celui rencontré par le flot 2 est RTT_2. Le lien d'accès pour le flot 1 (celui ayant S_1 pour source et D_1 pour destinataire) a un temps de propagation T_1. Sauf précision, les valeurs de RTT_1 et RTT_2 ne prennent en compte que les temps de propagation sur les liens et pas le temps de mise en file d'attente. Le temps de propagation sur le lien (S_1, R_1) est noté T_1.

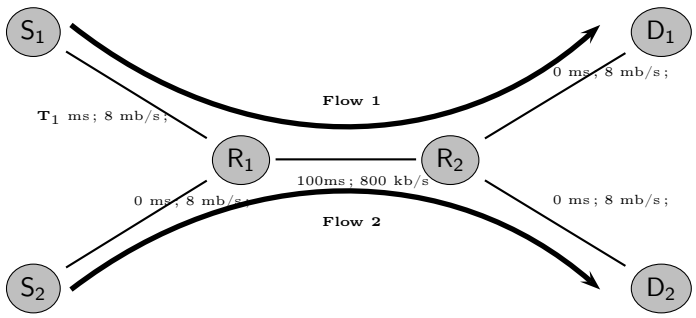

FIGURE 1 – Topologie du réseau mis en place dans le fichier `ns-tme.tcl`.

a. Donnez la valeur de RTT_2 et l'expression de RTT_1. Soit α un coefficient tel que $RTT_1 = \alpha.RTT_2$. Exprimez T_1 en fonction de RTT_2 et α.

b. Tracez sur un graphe la courbe du débit écoulé et du taux de perte de chaque flot en fonction de α avec α variant de 1 à 1.1 avec un pas de 0.02. Vous pourrez vous aider du script `sh launchMultipleExp.sh runAlphaTpDropDT`. Le partage de la bande passante est il équitable entre les deux flots ? Si oui, dans quelles conditions ? Quelle est la raison de la différence de débit ?

c. Retenez un cas où le partage est le plus inéquitable. Introduire un temps aléatoire de traitement des acquittements (paramètre `overhead`) de valeur maximum de 0.1 seconde. Vous pourrez vous aider du script `sh launchMultipleExp.sh runOverhead`. Que remarquez vous ? Pourquoi ? A la lumière de cette dernière expérience, expliquez le phénomène que vous avez constaté à la question 1.

d. Reprenons le modèle de la figure 1 avec $T_1 = 0$ et en indiquant un démarrage du flot 2 avec un retard de 5 secondes sur le flot 1. Vous pourrez vous aider du script `sh launchMultipleExp.sh runStudyDropTail`. Tracez la fenêtre de congestion de TCP (`cwnd`) de chaque source en fonction du temps. Pour ce faire vous pourrez utiliser le script `printQueueSizeCwndAndTp.sh` qui traite le fichier `cwndAndQueue.tr`. Ce fichier est généré par `ns` via l'observation périodique des valeurs de la cwnd des flots 1 et 2 ainsi que la taille de la file d'attente du lien $(n_0 - n_1)$. Veillez à mettre une valeur adéquate pour le paramètre `overhead`. Toujours à l'aide du script `printQueueSizeCwndAndTp.sh`, tracez l'évolution de la longueur de la file d'attente en fonction du temps. A l'aide de ces deux graphes, que remarquez vous ? Le goulot d'étranglement est il toujours en train d'écouler du trafic ? Quelle est la taille moyenne de la file d'attente ? La file d'attente déborde-t-elle ?

Pour conclure, à partir de vos observations, indiquez les effets indésirables du DropTail ?

Exercice 2 — Discipline d'attente - Front Drop

De nouvelles disciplines d'attente ont été proposées afin de remplacer le Droptail. Cette étude vise à mieux comprendre l'avantage de ces disciplines d'attente pour le service best-effort et comment elles peuvent améliorer ce dernier. Pour évaluer les améliorations, nous allons retenir le modèle de simulation représenté par la figure 1 avec $T_1 = 0$ par défaut. Les métrique que nous retiendrons seront le débit moyen écoulé et l'équité entre les flots. Une première idée a été de déporter les pertes de l'arrière de la file à l'avant de la file. Changez la politique de perte de Droptail de la file d'attente du goulot d'étranglement de manière que les pertes s'effectuent par l'avant. La politique de perte s'indique par le booléen drop_front (dans le script `ns-tme.tcl`, front drop se spécifie via `Queue/DropTail set drop_front_ true`.

a. Tracez sur un graphe la courbe du débit écoulé et du taux de pertes de chaque flot en fonction de α avec α variant de 1 à 1.5 avec un pas de 0.02. Vous pourrez vous aider du script `sh launchMultipleExp.sh runAlphaTpDropFD`.

b. La perte des paquets par l'avant de la file permet elle de supprimer l'effet de phase observé avec `droptail` ?

c. La répartition des ressources est-elle aussi inéquitable lorsque les flots ont des RTT différents ?

d. Est-ce nécessaire d'utiliser une valeur de l'`overhead` supérieure à zéro ?

e. De la même manière que dans l'exercice précédent, étudiez l'évolution de la taille de la file d'attente et de la fenêtre des flots lorsque $T_1 = 0$.

f. Au regard de vos experimentations et des notions vues durant le cours, critiquer les avantages et inconvénients de `frontdrop` par rapport à `droptail`.

Exercice 3 — Discipline d'attente - RED

L'IETF recommande le déploiement de la discipline d'attente RED (*Random Early Detection*) dans les routeurs de l'Internet [RFC2309]. RED vise à corriger les défauts de Droptail pour aider le contrôle de congestion de TCP. RED repose sur l'idée qu'il ne faut pas attendre que la file d'attente d'une interface de sortie d'un routeur soit pleine pour que les sources détectent la congestion et commencent à agir. Il faut anticiper l'apparition de la congestion lorsque la charge est forte afin d'éviter le débordement de la file d'attente et de jeter les paquets par blocs. RED déduit le niveau de congestion par la taille moyenne de la file d'attente (avg) et vérifie qu'elle se trouve entre un seuil minimum (min_{th}) et un seuil maximum (max_{th}). Si c'est le cas, un paquet qui arrive est jeté avec la probabilité $p = \frac{avg - min_{th}}{max_{th} - min_{th}} \times maxp$, avec $maxp$ la probabilité maximal de jeter un paquet. Tous les paquets qui arrivent quand $avg > max_{th}$ sont jetés. La taille moyenne de la file d'attente est calculée selon une moyenne exponentielle (EWMA). Le paramètre avg est initialisé à 0 puis à chaque arrivée de paquet il est actualisé comme suit :

$$avg = (1 - w_q) * avg + w_q * q$$

1. Le manuel d'utilisation est inclus dans l'archive qui accompagne ce TME et il est également disponible en ligne : `http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf`

où q est la taille réelle de la file d'attente et w_q est la pondération. RED souffre d'un gros défaut qui porte sur la configuration de ses paramètres. Les paramètres de configuration de RED s'indiquent de la manière suivante :

```
Queue/RED set thresh_ 5 # param min_th
Queue/RED set maxthresh_ 15 # param max_th
Queue/RED set q_weight_ 0.001 # param w_q
Queue/RED set cur_max_p_ 0.2 # param maxp

Queue/RED set adaptive_ 0
Queue/RED set gentle_ false

Queue/RED set bytes_ false
Queue/RED set queue_in_bytes_ false
Queue/RED set mean_pktsize_ 1000
Queue/RED set idle_pktsize 200
            ...
```

On considère le même modèle que dans la question précédente sauf que la discipline d'attente au niveau du goulot d'étranglement est maintenant RED.

a. Comparez les performances de RED par rapport à Droptail et FairDrop en fonction de α. Vous pourrez vous aider du script `sh launchMultipleExp.sh runCompareRed`.
b. RED permet il une amélioration de la capacité comparé à DropTail (débit total écoulé) ? Ce débit total est-il impacté par l'augmentation de la dissymétrie de RTT ?
c. La dissymétrie de RTT à telle un impact fort sur l'équité (c.f. index de Jain) ?
d. L'équité est elle améliorée par rapport à DropTail et FairDrop ?
e. Observez l'évolution de l'occupation de la file d'attente et de la `cwnd`.
   — L'évolution de la cwnd des flots TCP est elle toujours synchronisée ? Pourquoi ?
   — La file d'attente varie t'elle autant qu'avec droptail ? Quelle est sa taille moyenne ?
   — Les conditions rencontrées par un flot de VoIP qui partagerait cette file d'attente sont elles meilleures qu'avec droptail ? Si oui pourquoi.
f. Est-ce possible d'améliorer les performances de RED en terme de capacité, d'équité, de variation de la taille de la file d'attente ? Si oui, effectuez des tests et donnez les valeurs des paramètres min_{th}, max_{th}, $maxp$ et w_q qui permettent d'obtenir de meilleur résultats.
g. Même questions lorsque ECN est activé. Portez une attention particulière au taux de perte lorsque ECN est activé. Faite une critique, notamment au regard des flots cours et des applications non congestion contrôlés (*e.g.* VoIP) qui partageraient ce goulot d'étranglement avec TCP. Faites une synthèse des améliorations apportées par RED et ses variantes et justifiez ces dernières.

Exercice 4 — Taille des buffers

Le but de cet exercice est d'étudier l'impact de la taille de la file d'attente sur la capacité de réseau et sur l'efficacité. Vous pourrez vous aider du script `sh launchMultipleExp.sh runQueueSize`.

a. Faites varier la capacité de la file d'attente du lien $(R_1 - R_2)$ de 2 à 200 paquets. Tracez pour chaque discipline d'attente, le débit total écoulé, le taux de perte et l'index de Jain.
b. Avec droptail, quelle est la taille de file d'attente minimum à partir de laquelle le débit écoulé est proche du maximum ?
c. Même question avec FrontDrop, RED et ses variantes.
d. Même question avec l'équité.
e. Quelle est la discipline d'attente qui permet d'obtenir des performances correctes en terme de débit écoulé et d'équité tout en ayant une file d'attente de faible capacité ?

1 Annexes

1.1 Environnement de simulation :

Vous pourrez utiliser le fichier `ns-tme.tcl` qui contient la configuration du scénario de simulation pour ns-2 [2]. Les paramètres de simulation qui peuvent être passés en arguments au script `ns-tme.tcl` sont listés dans le tableau 1.1.

2. Le manuel d'utilisation est inclus dans l'archive qui accompagne ce TME et il est également disponible en ligne : `http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf`

Paramètre	Valeurs	Val par défaut	Signification
`delayX`	Xms	$0ms$	le délai de propagation sur le lien (S_1, R_1).
`queue`	{ `droptail` ; `red` }	`droptail`	Type de file d'attente.
`frontdrop`	{ `true` ; `false` }	false	Agit comme frontdrop si la queue est de type `droptail`.
`overhead`	[0 ;1]	0.0	Le paramètre d'overhead pour TCP *i.e.* le temps aléatoire attendu avant l'envoie des segments TCP.
`setecn`	{ true ; false }	`false`	Active l'option ECN sur les files d'attente et sur TCP.
`outfile`	-	`out.tr`	Le nom du fichier de trace généré par le simulateur.

TABLE 1 – Liste de paramètres, valeurs et valeurs par défaut pour le arguments qui peuvent être passés au script `ns-tme.tcl`.

Pour lancer une simulation utilisez la commande `ns ns-tme.tcl arguments`. Lisez attentivement le contenu du script `ns-tme.tcl` afin de déterminer les arguments qu'il vous semble nécessaire de spécifier. Lors de chaque simulation, les résultats sont stockés dans un fichier de trace [3] nommé `out.tr` qui conserve les principaux évènements *e.g.* réception d'un paquet sur un lien, mise en file d'attente, réception par le destinataire (...). Afin de faciliter l'analyse du fichier `out.tr`, le script `getStatsFromFlows.sh` calcule la quantité de données envoyées, le débit moyen écoulé et le taux de perte pour chaque flot et pour l'agrégat. Lisez le script afin de déterminer comment sont formatées les données qu'il affiche à l'écran. Le lancement des simulations et le stockage des résultats peuvent être automatisés via le script `launchMultipleExp.sh`. Pour afficher les résultats sous forme de courbes et graphiques, vous pourrez utiliser l'outil `gnuplot` [4]. Le script `computeStat.sh` est appelé par `launchMultipleExp.sh` et se charge de l'affichage des résultats sous forme d'image `png`.

3. Les format de trace est disponible à l'adresse : `http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats`.
4. Le manuel de gnuplot est inclus dans l'archive qui accompagne ce TME. Il est également disponible à l'adresse `http://www.gnuplot.info/docs_4.6/gnuplot.pdf`